Challenges and Opportunities Related to the Design, Deployment and, Operation of Web Services

Kostas Kontogiannis

National Technical University of Athens
Dept. Of Electrical & Computer Engineering
Athens, Greece
kkontog@softlab.ntua.gr

Abstract

Web Services have been proposed almost a decade ago as an implementation technology for Service Oriented Architecture. Web Service technologies have since been adopted by many users as a vehicle to build such service provision-based software systems. Despite their widespread adoption, Web Services still pose significant challenges as well as, opportunities both to the Information Technology community and, to Business community. The challenges deal with engineering, and adoption issues while the opportunities deal mostly with business and operations issues. In this paper, we first discuss the state of the art in the area of Web Services, and then we proceed on identifying challenges and opportunities related to designing, implementing, operating and maintaining such systems. Finally, we present emerging technologies that we believe may play a significant role in implementing and deploying the next generation Web Services and Service Oriented Systems in general.

1. Introduction

Web Services have been introduced as a particular implementation technology for building Service Oriented Architecture systems. Service Orientation has been proposed over the past two decades as the conceptual and architectural means for designing and implementing large scale distributed systems. These systems are composed of various run-time components that offer specific functionalities in the form of services to other components. Service Orientation has taken over the past two decades different forms and shapes. Having started as a simple client-server two-tier architecture, it has evolved to a form of distributed components that are encapsulated in object wrappers which offer unified means of invoking such wrapped components. Remote Method Invocation (RMI) and CORBA fall in this category of distributed object technologies.

A further evolution of these technologies along with the emergence of simple, yet highly practical Internet application-layer protocols and standards such as HTTP and XML, led to an infrastructure that allows for remote systems to be encapsulated by specialized wrappers which are able to handle HTTP connections and dispatch service requests from client processes to the wrapped systems. This infrastructure that utilizes open Internet Web-based protocols and allows for loose interconnection of systems to compose large distributed applications and possibly enact and implement specific business activities and processes, is referred to as Web Services.

Since the inception of Web Services, a number of supporting protocols have emerged over time. These protocols allow for handling security, handling transaction management, describing services, and for specifying quality of service characteristics and service level agreements, to name a few. Nowadays, Web Services as a technology have matured to a point where technologies, patterns and best practices have started to emerge and being applied in real life large-scale business and commerce applications.

The architectural abstraction that is relevant to Web Services is an architectural style referred to as Service Oriented Architecture or SOA. Service Oriented Architecture is based on the basic concepts of publish-subscribe protocols and service repositories. The fundamental concepts behind SOA (Figure 1) fall in three main categories.

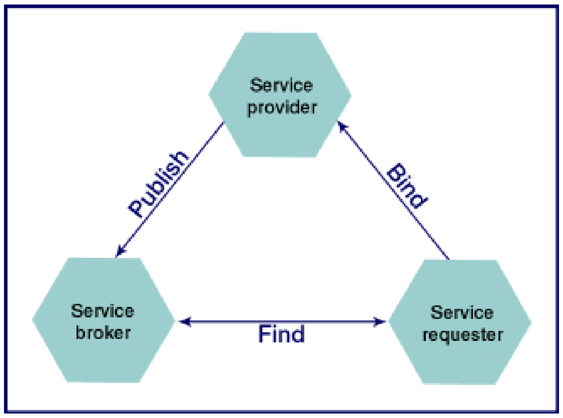

Figure 1. Service Oriented Architecture Style

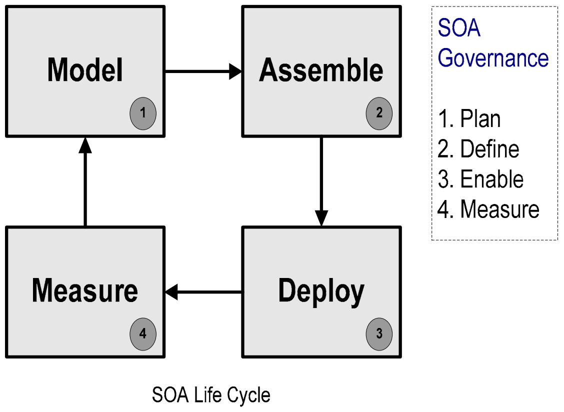

Figure 2. SOA Life Cycle and Governance

The first category deals with protocols that allow the description and specification of services. These descriptions can be *syntactic* to aim for the specification of the signature and the interface of each service, or semantic that is to aim for the specification and denotation of quality characteristics, type, as well as category and behavior of each service based on some ontological classification.

The second category deals with the specification of repositories that allow for service descriptions to be stored and discovered by client processes. Such service repositories act as yellow pages or more technically, as binders for the system. The fundamental idea is that a service can be selected among similar ones if it best matches some invocation criteria and objectives that a client process requires. In this respect, a service handler is returned to the client with all the necessary information on how a service can be invoked. The service selection process and logic may vary according to the application domain and implementation. To date, there is very limited use and dissemination of such public service repositories. The majority of such repositories are within corporate networks.

The third category deals with the invocation of a service once a handler is returned to the client process. The invocation is based on protocols that allow the request to be transmitted and handled by utilizing open protocols such as SOAP or, proprietary ones when specific requirements or objectives must be met.

A further aspect of service orientation and in particular Web Services is the ability to compose complex services from simpler ones. This is achieved by choreography or orchestration specifications denoted in languages such the Business Process Enactment Language (BPEL), and the Business Process Modeling Notation (BPMN).

Service Oriented Architecture as an architectural style that can be used for implementing Web Services poses unique opportunities and challenges in today's information world. First, it aligns in a natural way Information Technology (IT) entities with business models. Second, it provides the enabling technology for making available through open protocols, legacy components as services. Finally, it allows for the creation of complex systems from simpler ones forming of what is known as Systems of Systems (SoS).

In this paper, we present an overview of the current state of the technology in the area of Web Services and we discuss challenges, opportunities and, emerging trends in this important area of Information Technology. The paper is organized as follows. Section 2 discusses the state of the art in the area of Service Oriented Computing and Web Services. Section 3 discusses Challenges and Opportunities. Section 4 discusses emerging trends and Section 5 provides the conclusion and an outlook for Web Services and services oriented computing technologies.

2. Outline of State of the Art and Practice

2.1 Service Computing Life Cycle

Based on best practices, IBM has defined a life cycle for SOA systems that is consistent with other work on SOA life cycle [1], [6], [11], [15]. This SOA life cycle, presented in Figure 2, consists of the following phases: modeling, assembly, deployment and management.

Modeling is the process of capturing business requirements, business goals and objectives, and transforming them into business process specifications—the *business model.* The modeling phase also includes the analysis of the model—"what if scenarios" applied to the business processes. The *Assembly* phase deals with the implementation issues: the business models are implementing by either reusing existing services or by creating new services. Functional testing is part of this phase. The *Deployment* phase includes resolving service dependencies, capacity planning, defining the hosting infrastructure, as well as system testing. The *Management* phase refers to the operational activities that keep the applications running as well as the measurement of IT and business performance indicators, logs and traces for auditing, and feedback for other phases of the SOA life cycle.

To ensure successful deployment of the SOA life cycle, it should be done in the context of what is known as SOA governance, as also shown in Figure 2 [2], [16]. SOA governance is the process of establishing the chain of responsibilities and communications, policies, measurements, and control mechanisms that allow people to carry out their responsibilities. SOA governance itself has a set of phases: plan, define, enable, and measure. The *Plan* phase documents the existing IT capabilities and defines a governance plan. The *Define* phase defines or modifies the governance processes and the governance infrastructure. The *Enable* phase deploys the governance mechanisms and infrastructure, as well as the policies. The *Measure* phase monitors the compliance with policies and the effectiveness of the governance. SOA governance typically includes policies and procedures, roles and responsibilities, design-time governance and runtime governance [5].

2.2 Design and Implementation

Service Oriented systems and Web Services have been well investigated over the past few years from the design and implementation perspectives and there has been already many mature middleware technologies for process management. Furthermore, wrapping technology and its implications have been well understood. The state of the art and in practice here deal with architectural styles such as three tier and multi-tier architectures, heterogeneous architectures as well as with protocols and frameworks such as SOAP, UDDI, WSDL, J2EE, and .Net to name a few.

2.3. Deployment and Operations

With respect to deployment and operations we are observing a number of interesting patterns and practices that have emerged over the past few years. First, we observe a new evolution paradigm that is based on a *perpetual beta* type of evolution model. In this respect, SOA systems undergo *Incremental Development* and *Continuous Evolution* that is applied even when the systems have already been in operation. This paradigm differs from the classic view where the development and maintenance phases are distinct. This poses new challenges and opportunities for the maintenance of such systems. Second, we observe a silo-based type of adoption and operation of such systems where services are offered only by a collection of pre-defined pre-selected services defying in this respect the concept of a service repository, a concept that has not caught up as the rest of SOA technologies did. Furthermore, we have not yet witnessed the evolution of these systems to a point where they form Systems of Systems. However, the computing needs and the availability of the appropriate hardware and networking resources make *certain* the formation of such Systems of Systems that are also referred to as Ultra Large Scale Systems [10]. Finally, we observe a shift towards, and the consequent emergence, of virtualization techniques that allow for the better utilization of software and hardware resources in high loads.

2.4. Cross Cutting Issues

In addition to the above, there is a number of related issues that are peripheral, yet important, to the deployment of Web Services. For these issues we observe a gradual and increasing activity that has taken different forms and shapes according to the area and application domain systems are deployed. These issues can be classified as cross cutting issues and deal with a) *governance* and *compliance* and more specifically with techniques and processes to model policy, risk, and trust, and to ensure that a service acts on requests that comply with claims required by policies; b) Social and legal Issues that are related to the deployment and use of services in different jurisdictions and; c) People Skills/Capital an area that is related to the analysis of skills required to develop, use, and maintain a service-oriented system. Services Science Management and Engineering (SSME) that is discussed in more detail below, is one growing area in this theme.

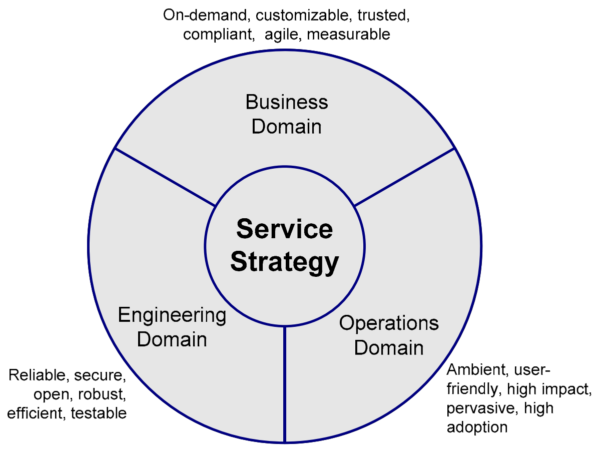
On-demand, customizable, trusted, compliant, agile, measurable
Business Domain
Service Strategy
Engineering Domain
Operations Domain
Reliable, secure, open, robust, efficient, testable
Ambient, user-friendly, high impact, pervasive, high adoption

Figure. 3 SOA Domains

3. Challenges

3.1 Business Strategy and Service Orientation

Web Services create new opportunities for the business world as they allow for the offering of a wealth of services over the Internet. The phenomenal growth of e-commerce and B2B transactions over the Internet has become the motivating factor for many corporations to consider Web Services and Service Orientation in general as a potential area of investment. However, Web Services can not, and should not, be implemented in a vacuum by considering only the technical / engineering challenges and issues. On the contrary, these systems have to be considered, designed, implemented and deployed only in par with a specific Business Strategy. The Business Strategy defines among other entities the Business Model for Service Orientation. In this respect, the Business Model defines the function of the applications and services with respect to a specific Business Strategy, while the corresponding Service Model denotes how this functionality is to be provided. Business Models for service oriented systems is affected by a number of constraints such as user constraints, technology constraints, laws, legislations, competitors, areas of service offerings, access rights to a service as well as, the computing platforms and the available infrastructure

Event though there is a significant work in the area of defining business strategies and business models, to date we have very limited guidelines and techniques to establish and document a business case for service orientation. This is due to a number of factors. First, technology is evolving rapidly and becomes a moving target both on budgetary terms, and on technical capabilities. Anticipating or forecasting the next technological innovation and its associated costs is not always easy, and this may complicate the formation of the long term business strategy decision making process.

Second, business needs may change over a short period of time as clients' needs and product offerings may also change, especially when services and products are offered on a world-wide scale to a diverse clientele.

Third, the revenue generating model for service oriented systems is not as clear as it is the revenue generating model that is related to services offered in the traditional sense. This is due to different economics of scale as well as, technology, training, operational and administration issues related to service orientation such as the need to develop models for contract pricing and negotiation in an on-demand service setting. Additional challenges include the investigation of the relation between IT metrics and business metrics as well as, methods, models and representations for assessing the effectiveness of services.

Furthermore, to date we have not yet established techniques for selecting which service strategy is best suited for any given business domain or a type of related domains. By the term service strategy we refer to the process and activities that tie together Business, Engineering, and Operations domains of a Service Oriented Architecture together, informing the decisions made in each of these domains, and reflecting the influence they have on each other (Figure 3). A Service strategy provides the cause-effect and impact links behind the decisions taken at the Business, Engineering and, Operations levels in an organization and it provides the common ground for the analysis of a service-oriented system that takes into account different perspectives and points of view.

Finally, another issue that posses challenges for the formation of a business strategy is how to map business processes to the appropriate services and how well a given service fits specific business needs. This includes techniques for service identification, analytic methods for service evaluation, techniques and processes for establishing relations between business and service models with patterns and stereotypes for roles and responsibilities of the involved stakeholders and models for organizational structures in SOA-environments.

3.2 Service Definition and Categorization

In order for services to be evaluated either for selection or compliance reasons, these must be described using a thorough and detailed specification mechanism. This specification mechanism must provide syntactic, semantic, and behavioral information about the service. Currently, Web Services are specified using a standard format namely the Web Services Description Language (WSDL). This formalism focuses mostly on the syntactic nature of the service interface and on details related to the points of service offering. Over the past few years standardized ontologies for specific application domains (e.g. banking, accounting, computer hardware, automotive) have started to emerge and this created the necessary momentum for the foundations of behavioral specifications of services. Formalisms such as DAML-OIL and OWL have become the de-facto standards in the community for the behavioral specification of services. In a similar way, Service Level Agreements (SLA) have emerged as a way for services to post specific verifiable guarantees about specific aspects of the quality of the offered service functionality. Even though, we have made significant progress towards the description of services, we have fallen short on defining standard and prescribed ways for denoting semantic and meta-data information about services. It is currently an active area of research that has even more interest when other types of service offerings such as REST compliant services are to be considered.

In many cases, service offerings must be adapted according to the context they are invoked. The invocation context is defined as the environment in which a service is to be deployed by keeping in mind that operations not always operate under ideal conditions. In this respect, the service model of the offered services must be adapted according to the specific requirements of the invocation context. These requirements may relate to client-side presentation logic of the results, QoS characteristics of the service, access rights, as well as invocation and messaging types (e.g. synchronous, asynchronous). Context aware service provision remains one of the challenges on implementing services that are widely adopted.

Finally, there is limited work to date on investigating techniques and processes to support the strategic reuse of components and services. This area can be considered of in two dimensions. The first dimension deals with techniques and programming models that allow the design of reusable components. The investigation of novel design patterns and the compilation of best practices constitute interesting challenges for the design and implementation of such reusable assets. The second dimension deals with the reuse of existing legacy components and services. In order to achieve such reuse of legacy components and services we must consider novel reengineering, migration and, integration techniques. System-wide slicing techniques, system-wide dependency models, impact analysis and wrapping techniques as well as, data and event mediation frameworks such as service buses are all challenge and interesting research areas in this domain.

3.3 Design and Implementation

It is a fact that Web Service technology has matured due to a collection of open protocols that allow for the description and invocation of remote components. Such open protocols include WSDL and SOAP. However, there is a growing number of other protocols that are also important but have not reached yet the level or wide adoption WSDL and SOAP have achieved. Such protocols deal with security, authentication, transaction management, and, monitoring to name a few. These protocols are referred to as WS-* protocols. The diversity of all these competing in may cases, protocols that have been introduced in these areas poses both a challenge and a threat. It is a challenge as these protocols aim to advance the state of the art. It is also a threat because as long as there is such a diversity in these protocols standards are more difficult to emerge, and therefore one of the fundamental principles of Web Services namely openness, is at stake. A challenge in this area is the investigation of how these protocols can be amalgamated so that standards can be proposed, or how mediators and transformers between the different specification formats, can be built.

The potential and the demand for more wide spread design and deployment of Web Services posses another challenge namely the need for highly qualified technical personnel who have the programming skills and software engineering knowledge to architect such systems. At this point in time, the community has realized that simply there is not enough skilled people power to design and implement the Web Services and the Service Oriented Systems in general, that we will need in the next few years to come. For this purpose, the research community has focused in two areas that pose interesting challenges and potential. The first area is the quest for a new programming model for Web Services and Service Oriented Architecture. The focus here has been in developing new programming structures that ease the burden of programming of such systems. In [3] such a programming model revolves around Part Types, Roles, Skills, Application Interfaces and Tools that support a) the provision of a simplifying abstraction that allows programmers to concentrate principally on business logic and not on programming details; b) the provision of uniform representation for messages that interact with services, technologies for creating and accessing business logic and finally; c) ESB technology and means related to facilitating auditing, logging, routing, adaptation of mismatched interfaces and security, at an enterprise-wide level.

Furthermore, Model Driven Architecture (MDA) and Model Driven Engineering (MDE) pose significant challenges and opportunities for facilitating the automatic or semiautomatic code generation for such systems from higher abstraction design and specification models. To this extend, challenges relate to the investigation of what models are appropriate and complete to denote Web Services at a point of detail where automatic or semiautomatic code generation will be possible, and what are the appropriate model transformation frameworks, and model co-evolution techniques so that models remain consistent when one or more of them change due to maintenance and evolution.

Other challenges related to the design and implementation of Web Services include the investigation of transaction management techniques for large scale systems with long running transactions, novel messaging and, invocation protocols, the investigation of the use and the role of CBSE in Service Oriented Systems as well as, techniques that allow the personalization and adaptation of services.

3.5 Testing, Monitoring and Diagnostics

Web Service application systems are by definition large scale, loosely coupled, distributed systems. As such, they pose significant challenges with respect to testing, monitoring and, diagnostics.

More specifically, with respect to testing these systems introduce new challenges related to integration, system, and regression testing, due to the loosely coupled components involved, and the nature of the transactions in such systems. Furthermore, acceptance testing is more diverse due to the wide variety of stakeholders and system users. In this respect we may consider testing challenges for Web Services falling in three areas a) Infrastructure Level Testing, where we aim to develop techniques to test and verify the correctness of SOAP messages, WSDL specifications, UDDI specifications and any other related to the application WS-* protocols; b) Application Level Testing, where we consider Web Services as components that require functional testing and transaction management testing; c) Global Dynamic Testing, where the objective is to test the composition, orchestration, versioning, monitoring as well as to perform regression testing and load/stress testing and finally; d) Business Level Testing where the objective is to test the conformance of the services against Service Level Agreements and other business process requirements.

With respect to monitoring, challenges include the use of log data from such large applications for error prediction. The point here is to store logs in data warehouses and possible to mine these warehouses to predict system errors or to identify potential threats or policy violations. In this context one could consider that such systems may be able to provide Business Intelligence solutions by monitoring mission and application critical components.

Another challenge deals with logging, monitoring and diagnosis not only at the IT level but also at the Business Process level. The premise here is that large business applications can be considered and analyzed at three levels of abstraction. The lowest level pertains to software components and source code. The next level pertains to interactions among components at the API or protocol level. The highest level of abstraction pertains to business processes. Business processes are not necessarily involve only software systems but may also involve humans.. To date, most logging frameworks focus on monitoring the software components and there are no adequate frameworks to monitor systems at the business process level. This requires the weaving of logs from the actual system, and input from the manual or human activated part of the processes. The objective is to provide information to managers and IT professionals with respect to the application as a whole by interpreting the behavior of the system with respect to the context and the business process it operates on. In [14] a framework for cross cutting monitoring between infrastructure, application and, business levels has been proposed as a potential solution to this problem.

Finally, another challenge area deals with developing logging and monitoring frameworks for Ultra Large Scale (ULS), Service Oriented systems. More specifically, a challenge area is to investigate an infrastructure whereby logging, monitoring, diagnosis and root cause analysis of such systems can be achieved in a scalable and adaptive manner. Scalability deals with the size and complexity of such systems, while adaptivity deals with the selective and incremental as-required logging according to customizable monitoring objectives. To date there is limited work on logging infrastructures that exhibit such adaptive behavior and are tailored for SOA type of applications.

3.6 Maintenance

During the last decade, the number of software applications that are implemented using multiple programming languages has increased considerably. When such applications are maintained, program comprehension and re-engineering techniques are required that can handle multiple languages. Multi-language (ML) systems pose a whole new range of challenges with respect of extracting information and modeling dependencies between the components. Some of these challenges relate to parsing, the utilization of different language constructs with different semantics and the diverse designs such multi-language system encompass. Furthermore, processing large volumes of information that can be extracted form ML systems requires the use of sophisticated tools and tractable algorithms.

Finally, the use of tools to analyze, maintain and evolve large Multi-language systems is bound to have an effect on the maintenance and evolution process models. Some questions that arise in such a Multi-language analysis and maintenance environment include the type of metrics or other quantifiable means to use in order to reason about the evolution process, to measure the impact of ML software on software maintenance and evolution and to establish a common measurement framework.

Another challenge area for the maintenance of Web Services has to do with the nature of the evolution cycle of these systems. Web Services and SOA systems in general do not follow a classic specify-design-test-deploy paradigm. These systems are under constant re-implementation and maintenance either to add new functionality, or port these systems to new platforms. Some refer to this type of operation as *perpetual beta*. To date we do not have developed the appropriate infrastructure and tooling to efficiently deal with this type of continuous iterative and incremental evolution model. Techniques for model synchronization, model transformation, model refactoring and model co-evolution may play an interesting and important role [9].

3.7 Security and Operations

Security has been always in the forefront of challenge areas for Web Services and has been well investigated at the infrastructure level (cryptography, authentication). In a nutshell, we may consider that the broad area of securing Web Services and SOA based systems falls in two major topics a) securing the integrity and confidentiality of messages and most importantly exchanged data and; b) ensuring a service acts on requests that comply with claims required by service level agreements and business policies. Areas of challenge here deal with developing techniques for ensuring secure messaging, techniques for validating policies, techniques for evaluating risk and techniques for ensuring trust on services and service vendors.

With respect to operations, areas of challenge include the topics of governance, compliance, stakeholder management, legal issues related to offerings of services beyond the boundaries of a specific jurisdiction as well as, training and education issues. Governance and compliance emerge as areas of particular interest to the technical community. Governance ensures that there is a chain of command and there is information available to identify who (or which system or service) does what. Compliance ensures that whatever service or action is performed is compliant with specified policies, constraints and processes. Particular technical challenges in this area include a) devising techniques and proposing frameworks for modeling and abstracting IT events (infrastructure / application layer) as well as modeling system properties and policies at the business level (business layer); b) investigating logging and monitoring (adaptive) as means to populate specific run-time behavior models of large software systems and; c) comparing and validating these run-time behavioral models against business process models, so that compliance analysis, auditing, as well as intelligent hardware and software resource allocation can be achieved.

4. Opportunities and Emerging Trends

4.1. Services Science

Services, and in particularly software services, have been identified as the largest sector of the economy in most industrialized nations and it is also

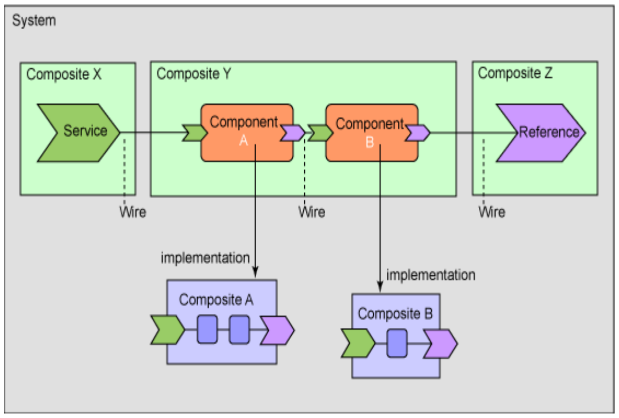

Figure 4. Service Component Architecture [12]

becoming the largest sector in developing nations as well. In this respect, over the past few years IBM has launched a global initiative that aims to investigate services not only from the technical and technology point of view but also as an interdisciplinary area that requires the expertise of various fields. More specifically, in [13] the term Service Science Management and Engineering (SSME) refers to interdisciplinary research and education that *aims to bring together ongoing work in computer science, operations research, industrial engineering, business strategy, management sciences, social and cognitive sciences, and legal sciences to develop the skills required in a services-led economy.* Several Universities and educational institutions have already started offering programs related to SSME. We believe this is an area of growth and opportunity in the years to come as it aims to bring together areas that provide different stakeholder views into building and operating Web Services and Service Oriented Systems in general.

4.2. Technology Frameworks

Service Component Architecture (SCA) is emerging as a strong initiative that is supported by a number of large software vendors such as BEA Systems, IBM, IONA Technologies Oracle, Red Hat Inc., SAP AG, Siebel Systems, Sun Microsystems, Sybase, TIBCO Software Inc to name a few. Service Component Architecture (Figure 4) [12] refers to a number of specifications that define a model for building Service Oriented Architecture system applications. The fundamental building blocks of SCA are Components that act as services or references for other components. Components may have associated policies and properties for testing, verification, validation and compliance purposes. The process of building an SOA system using SCA is split into two phases a) the implementation phase of service components which defines the services and b) the assembly phase that connects collections of components to build business applications, through a wiring mechanism. The strengths of the SCA are first, that SCA decouples service implementation and service assembly and second, components utilize the minimum API from the underlying middleware and it supports service implementations written in different programming languages. We believe that this technology area provides a wealth of opportunities especially for the design and development of supporting tooling for the implementation, monitoring and, versioning of SCA compliant systems.

Java Business Integration (JBI) is another specification of interest that poses new opportunities. It has been developed under the Java Development Process initiative and it provides a pluggable architecture for a container that hosts service producer and consumer components. Services connect to the container via binding components (BC) or can be hosted inside the container as part of a service engine (SE). The underlying services model is based on WSDL 2.0 while the message delivery mechanisms is based on the normalized message router (NMR) that delivers normalized messages using the Message Exchange Patterns of WSDL 2.0. Similarly to the above, opportunities in this area involve the design and development of support tooling for JBI related to implementation, monitoring and, versioning of JBI system.

REST and Web Services have been so far identified as complementary and diverse technologies. REST [4] stands for Representational State Transfer and it describes an architecture style of networked systems. REST is based on the premise that the invocation of a service can be accomplished by the use of resource identifiers and well formed URI descriptions. REST provides a lightweight yet powerful style for implementing SOA systems. On the other hand, issues relating to different types of messaging, versioning, and transaction management are not as efficient in REST as when using Web Services. The opportunity in this respect lies on the investigation of the relationship between REST and Web Services as well as the design of frameworks for bringing closer and utilizing better combined Web Services and REST services. This can be achieved possibly through extensions and the use of SCA or other frameworks such as JBI.

4.3. Corporate Mashups

Mashups are defined as digital media files that integrate different content such as text, graphics, audio, video and animation that is drawn from pre-existing sources, to create a new derivative work. Mashups have been very popular in social networking sites and for Digital mashups represent a new way of re-using of existing digital content and integrating this content with ease. Even though Mashups have been proposed for non-programmers, one could envision a simplified programming model that could leverage SOA principles and Mashups' ease of use. In particular, an area of opportunity is the design of frameworks that allow for corporate users (e.g. managers, executives) to compile quickly new Mashup applications from a pool of corporate services. These services may have to do with decision making support, business intelligence, process management, and governance. We believe that this is an area where existing popular technology can be transferred with the appropriate adaptation to the corporate world.

4.4. Software as a Service

The traditional view of software is that it is offered for purchase to users by software vendors. The software becomes a corporate asset and it is owned by the client. However this traditional view of software use has recently been challenged from successful vendors such as SalesForce. The new view is that software is rented as a service and it is paid according to the level of use. This has initiated a new emerging area referred to as Software-as-a-Service (SaS). This topic by itself is very intriguing, but when it is combined with the concepts of Service Oriented Architecture and Web Services becomes an interesting emerging opportunity. To date, we have only limited experience on how to utilize SOA frameworks to support SaS.

In this respect, we believe that the investigation of novel technologies, including virtualization, to identify how one could efficiently deploy SaS utilizing SOA principles is an emerging field that poses significant research challenges and commercial opportunities.

4.5 Adaptive and Autonomic Computing

In the recent years IBM Research has launched a new initiative that has been identified as Autonomic Computing. The fundamental principle behind these systems is an infrastructure that allows for self management. Self management has several facets and namely *self-configuration* that allows for the automatic configuration of components; *self-healing* that allows for the automatic discovery, and correction of faults; *self-optimization* that allows for the automatic monitoring and control of resources to ensure the optimal functioning with respect to the defined requirements and; *self-protection* that allows for the proactive identification and protection from arbitrary attacks [7]. In this respect an emerging area of opportunity is the design of frameworks that allow for adaptive and autonomic service oriented systems to be built. More specifically, we believe that concrete areas of research as well as practical opportunities exist in the investigation of novel techniques that either extend existing WS-* protocols so that a level of autonomic behavior in existing Web Services can be achieved or extend and utilize emerging frameworks such as SCA to build the next generation service oriented autonomic systems.

5. Conclusions and Outlook

Service Orientation has been proposed as a major driving force for offering software components as services over a networked infrastructure. Given the fact that the services sector and in particular the software services sector is among the highest growing sectors both in the industrialized world and in the developing world, we find ourselves facing a great opportunity and challenge. To better channel our research efforts, we should attempt to reflect upon our progress to date and recognize how our efforts and results build on each other, and to identify – and potentially prioritize – the areas that we still need to investigate.

In this paper, we discussed challenges and opportunities behind service orientation and Web Services. The challenges have been classified in seven key areas namely Business Strategy and Service Orientation; Process and Lifecycle; Service Definition and Categorization; Design and Implementation; Testing, Monitoring and Diagnostics; Maintenance and finaly; Security and Operations. Similarly, the opportunities and emerging trends have been classified in five areas namely Services Science Management and Engineering; Technology Frameworks; Corporate Mashups; Software as a Service; and Autonomic Computing.

We believe that the outlook for service orientation and Web Services is very positive and with it come the challenges to support this paradigm, as well as the opportunities for new research.

About the Author

Kostas Kontogiannis is an Associate Professor at the Department of Electrical & Computer Engineering at the National Technical University of Athens, on leave from the University of Waterloo, Canada. He is holding a Ph.D. degree in Computer Science from McGill University, Canada. He is currently working in the areas of software evolution, software systems integration and, software monitoring. Kostas has been the recipient of three IBM University Partnership Awards, a recipient of a Canada Foundation for Innovation (CFI) Award and, a former member of the IEEE Distinguished Visitors Program.

References

[1] Borck, J. *Planning an SOA: Gathering Around the Drawing Board*. Infoworld. May 2006. http://www.infoworld.com/article/06/05/08/77665_19FEsoalife2_1.html?s=feature

[2] Brown W. and Cantor, M. *SOA Governance: How to Oversee Successful Implementation through Proven Best Practices and Method*. ftp://ftp.software.ibm.com/software/rational/web/whitepapers/10706900_SOA_gov_model_app_v1f.pdf

[3] Ferguson D., Stocton M., *Introduction to the IBM SOA Programming Model, http://www.ibm.com/developerworks/webservices/library/ws-soa-progmodel/*

[4] Fielding, R T. Taylor, R. N., *Principled Design of the Modern Web Architecture, ACM Transactions on Internet Technology (TOIT) (New York: Association for Computing Machinery)* ***2*** *(2): 115–150,*

[5] Gold-Bernstein, B. and So, G. *Integration and SOA: Concepts, Technologies and Best Practices*.

[6] High, R., Kinder, S., and Graham, S. *IBM's SOA Foundation: An Architectural Introduction and Overview*. http://download.boulder.ibm.com/ibmdl/pub/software/dw/webservices/ws-soa-whitepaper.pdf

[7] *An Architectural Blueprint for Autonomic Computing* http://www-03.ibm.com/autonomic/

[8] Java Business Integration http://java.sun.com/integration/

[9] Mens T., Van Der Straeten R., On the Use of Formal Techniques to Support Model Evolution. http://idm.imag.fr/idm05/documents/18/P18.pdf

[10] Notrthrop L. et al., *Ultra-Large-Scale Systems: The Software Challenge of the Future*. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2006.

[11] Rodriguez, J. *New Rules Govern SOA Lifecycle*. http://www.looselycoupled.com/opinion/2005/rodri-rules-gov0701.html

[12] *Service Component Architecture* http://www.ibm.com/developerworks/library/specification/ws-sca/

[13] IBM Corporation. *Services Sciences, Management and Engineering,* 2006

[14] Traverso P.. *Agree of Change! Making Services to Evolve.* Keynote talk, IEEE International Conference on Software Maintenance 2007, Paris France.

[15] Veryard, R. *The SOA LifeCycle*. CBDI. August 2004.

[16] Windley, P. *SOA Governance: Rules of the Game.* InfoWorld. January 2006. http://www.infoworld.com/pdf/special_report/2006/04SRsoagov.pdf